
Algoritmos II




Tuplas

Prof. Fernando Del Moro

Tuplas

Assim como os vetores, tuplas tem o objetivo de armazenar múltiplos valores em apenas uma (1) variável;

De mesma forma, são indexados ou seja, seus elementos podem ser acessados através de índices;

Mas apenas acessados; 

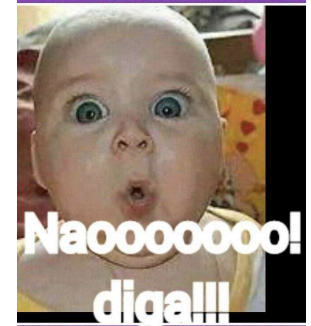
Tuplas

Diferente dos vetores, as tuplas:

Uma tupla é uma lista imutável de elementos, o que significa que não pode ser alterada de forma alguma depois de criada.



Tuplas



Por não permitirem mudanças elas são caracterizadas como "imutáveis" ou "constantes"

Variáveis permitem modificação enquanto constantes são imutáveis

Menos comuns que os vetores, são utilizados quando os valores inseridos não podem/devem ser modificados

Sua característica de imutabilidade dá mais segurança ao desenvolvedor e maior proteção ao código



Quando utilizar as Tuplas

Normalmente o uso das tuplas é indicado para armazenar dados heterogêneos ou armazenar informações que não serão alteradas.

Por exemplo, os meses, as estações do ano ou as vogais do alfabeto são informações que não serão alteradas com facilidade.

Sendo assim, esses são elementos que podem ser colocados em tuplas sem problemas.



Como criar tuplas em Python

Existem diversas maneiras de construir tuplas em Python, mostradas abaixo:

Par de parênteses para tupla vazia: ()

Vírgula à direita para identificar uma tupla *singleton*: a, ou (a,)

Itens separando por vírgulas: (a, b, c) ou a, b, c

Usando o tuple () *built-in*: tuple()

Como criar tuplas em Python

Exemplos:

vazia = ()

single = ('a',)

normal = (1,2,3,4,5)

tupla = tuple('algoritmos')

ATENÇÃO! Utilizar Parênteses
Tuplas utilizam "(" e ")"

ATENÇÃO 2! Precisa utilizar a ','
senão ele vai achar que é uma
String simples e não tupla

ATENÇÃO 3! Essa é a utilização
padrão. Parênteses não são
obrigatórios mas aconselhável utilizar

ATENÇÃO 4! Cria uma tuplas com as
letras



Como criar tuplas em Python

Assim que a tupla é criada e são definidos os elementos, não é mais possível alterar, incluir ou remover valores

Tuplas são "imutáveis"



Operações sobre tuplas

Como as tuplas não podem ser modificadas as únicas operações possíveis são **percorrer**, **testar** ou **apagar** uma tupla

O processo de percorrer uma tupla se assemelha muito ao percurso de um vetor, mas podemos mais....



Operações sobre tuplas

A forma normal de percurso que se assemelha ao uso em vetores

```
for cont in range(0, len(tupla)):  
    print(tupla[cont])
```

Operações sobre tuplas

Uma forma diferente de "for" chamada "for...each" onde os itens são iterados por uma variável

```
variável ← for item in tupla:
           print(item)
           → utilizada para acessar
              cada elemento da tupla
              que será iterada a cada
              loop
```



Operações sobre tuplas

A instrução *for...each* não utiliza índices para identificar cada elemento. Isso é feito pelo "for" que inicia a variável indicada no primeiro valor e a cada *loop* atribui a variável o próximo valor, até chegar ao final

for...each não tem acesso aos índices

Operações sobre tuplas

A instrução `for...each` não tem acesso direto aos índices pois isso não é sempre necessário

Se for realmente necessário é possível utilizar o *enumerate*

índice, variável

```
for cont, item in enumerate(tupla):  
    print(f'o item{item} no indice {cont}')
```

tupla

possível utilizar o índice e
a variável



Operações sobre tuplas

Outra operação possível sobre tuplas é o "fatiamento" dos valores quando eles forem mostrados

Fatiar significa que ao invés de mostrar toda a estrutura é possível mostrar apenas uma parte (fatia), sem estruturas de repetição

Todas utilizam alguma versão de [indice:indice]



Operações sobre tuplas

Exemplos:

[:] - mostra todos(desnecessário)

[:indice] - do início até <indice> (<indice> não é mostrado)

[indice:] - de <indice> até o final

[indice : indice] - do indice inicial até final (final não é mostrado)



Operações sobre tuplas

Outros exemplos:

`[-1]` - mostra o último

`[-2]` - mostra o penúltimo

`[-2:]` - do penúltimo até o final

`[:-2]` - do início até o penúltimo (penúltimo não é mostrado)



Outras funções sobre tuplas

`sorted(tupla)` - retorna a tupla ordenada mas não altera a original (lembre que é imutável)

`len(tupla)` - número de elementos da tupla

`tupla.count(elemento)` - retorna o número de ocorrências de <elemento>

`tupla.index(elemento)` - retorna o índice da primeira ocorrência de <elemento>

`tupla.index(elemento, indice)` - retorna o índice da primeira ocorrência de <elemento> a partir de <indice>



Outras funções sobre tuplas

`del(tupla)` - apaga a tupla da memória. OBS: não é apagado um valor e sim a tupla toda. Ela deixa de existir na memória

`+` - soma duas tuplas, gerando uma terceira, resultado da concatenação



Exercícios

1) Crie um programa que tenha um tupla preenchida com os valores por extenso de 0 a 15. Peça ao usuário um número de 0 a 15 e mostre o valor por extenso. Verificar limites de digitação.

2) Faça um programa que leia 4 valores inteiros e coloque-os em uma tupla. Mostre:

- Quantas vezes apareceu o número 9
- Em que posição está o número 3
- Quais foram os pares



Exercícios

3) Crie uma tupla preenchida com a tabela de classificação (em ordem de classificação) dos times do campeonato brasileiro da série B de 2023.

Mostre:

- Os 4 primeiros colocados(zona de acesso)
- Os 4 últimos colocados (zona de rebaixamento)
- Lista dos times em ordem alfabética
- Em que posição terminou o Criciúma E.C.



Exercícios

EC Vitória
Juventude
Criciúma
Atlético-GO
Novorizontino
Mirassol
Sport Recife
Vila Nova
CRB
Guarani

Ceará SC
Botafogo SP
Avaí
Ituano
Ponte Preta
Chapecoense
Sampaio Corrêa
Tombense
Londrina
ABC